PAT-NO: JP411203145A
DOCUMENT-IDENTIFIER: JP 11203145 A
TITLE: INSTRUCTION SCHEDULING METHOD
PUBN-DATE: July 30, 1999

INVENTOR-INFORMATION:

NAME	COUNTRY
MORIKAWA, NAOTO	N/A

ASSIGNEE-INFORMATION:

NAME	COUNTRY
HITACHI LTD	N/A

APPL-NO: JP10004056
APPL-DATE: January 12, 1998

INT-CL (IPC): G06F009/45 , G06F009/38

ABSTRACT:

PROBLEM TO BE SOLVED: To simplify scheduling which considers the effect of out-of-order execution by delaying the dispatch of an instruction whose pipeline length is long and by scheduling so that an instruction whose pipeline length is long may undergo out-of-order execution.

SOLUTION: A processor description table 20 is read. The purport that a processor can simultaneously execute two instructions is shown in a table 21 of the table 20 and an execution time that corresponds to the length of an instruction pipeline of each instruction and use latency of subsequent instructions are described on a table 22. If execution time of the subsequent instructions is long, the use latency of the instruction is increased more than the conventional manner that reflects actual processor operations. The time from each instruction dispatch to execution start and the time from execution end till completion are suppressed to be short and an instruction string is scheduled so that a time when each instruction occupies hardware resources may be short.

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号

特開平11－203145

(43)公開日　平成11年(1999)７月30日

(51)Int.Cl.[6]		識別記号	FI		
Ｇ０６Ｆ	9/45		Ｇ０６Ｆ	9/44	３２２Ｆ
	9/38	３１０		9/38	３１０Ｘ

審査請求　未請求　請求項の数１　ＯＬ　（全８頁）

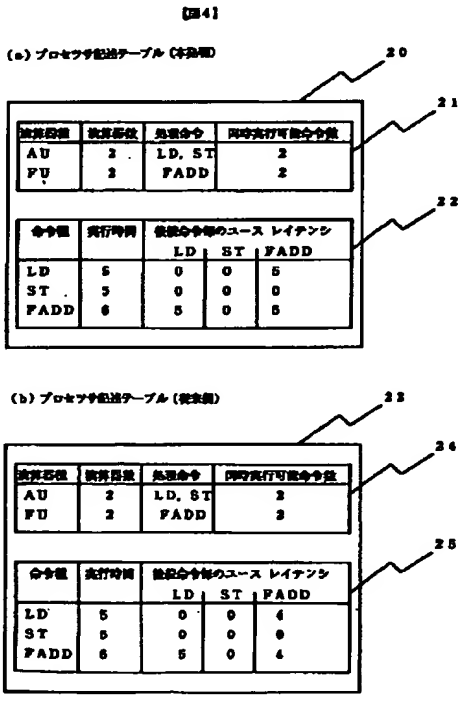
(21)出願番号　特願平10－4056

(22)出願日　平成10年(1998)１月12日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地
(72)発明者　森川　直人
神奈川県秦野市堀山下１番地　株式会社日立製作所汎用コンピュータ事業部内
(74)代理人　弁理士　武　顕次郎

(54)【発明の名称】　命令スケジューリング方法

(57)【要約】
【課題】　命令パイプライン長が一定値でないアウトオブオーダ実行を行うパイプライン計算機に対して、ディスパッチから実行開始までの時間及び実行終了からコンプリートまでの時間を短くおさえ、各命令がハードウエアリソースを占有する時間が短くなるように命令列をスケジュールする。
【解決手段】　マシン記述テーブルにおいて、後続命令のパイプライン長が長い場合のユースレイテンシを実際の値より大き目の値に設定してスケジュールすることによりパイプライン長の長い命令のディスパッチを遅らせ、その命令のアウトオブオーダ実行を誘発し、それにより後続命令のコンプリートのタイミングを早める。
（ｂ）は従来技術のものであり、本発明は、（ａ）に示すように、後続命令のパイプライン長が長いＦＡＤＤ命令のユースレイテンシを従来技術の場合より長く設定している。



1

【特許請求の範囲】

【請求項1】　命令列のアウトオブオーダ実行を行うパイプライン計算機の目的プログラムを生成するコンパイラにおける命令スケジューリング方法において、命令を実行する複数の演算器のパイプライン長がそれぞれ異なっている場合に、先行命令のターゲットデータをパイプライン長の長い後続命令が利用する場合のユースレイテンシを実際より大きく設定することにより、パイプライン長の長い命令のディスパッチを遅らせ、パイプライン長の長い命令をアウトオブオーダ実行させるようにスケジューリングすることを特徴とする命令スケジューリング方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、最適化コンパイラにおける命令スケジューリング方法に係り、特に、命令の実行順序がプログラムの命令列の順に行われることを保証しないアウトオブオーダ実行を行うパイプライン計算機上でのプログラムのハードウエアリソースの利用効率を向上するために有効なコンパイラにおける命令スケジューリング方法に関する。

【0002】

【従来の技術】基本ブロック内の命令スケジュール方法に関する従来技術として、例えば、アイ・ビー・エム ジャーナル リサーチ・アンド・デベロップメント 38巻5号（1994年9月）557頁―（Blainey,R.J. Instruction scheduling in the TOBEY compiler, IBM Journal Research and Development 38:5 (September1994),577.）に記載されたリストスケジューリング法と呼ばれる方法が知られている。

【0003】前記従来技術によるリストスケジューリング方法は、基本ブロック内の命令列に対し、各命令間のデータ依存関係とハードウエアリソースの競合とによる制約の下で、目的プロセッサ上での実行時間が最短となるように命令列を並べ替えるというものである。そして、前記従来技術は、演算器数、命令毎の実行時間、ユースレイテンシ等のプロセッサ固有の情報がテーブル形式（プロセッサ記述テーブル）で与えられ、命令実行の開始時間について考慮されているが、各演算器のパイプライン長、すなわち、命令の実行時間のばらつきによるコンプリーションの遅れが考慮されていない。また、前記従来技術は、コンペア命令とそれをユースする分岐命令のユースレイテンシを大目に設定し、これにより、分岐予測ミスによる性能劣化を防止するようにしている。

【0004】

【発明が解決しようとする課題】一般に、複数の演算器を使用して複数の命令を並列実行するスーパスカラ・アウトオブオーダ実行のプロセッサは、命令列をインオーダで各演算器にディスパッチし、各演算器上でアウトオブオーダで命令を実行した後、インオーダでコンプリートする。そして、演算の実行に使用するバッファ、レジスタ等のハードウエアリソースは、命令列のディスパッチ時に確保され、コンプリート時に解放される。

2

【0005】前述した従来技術は、命令のパイプライン長にばらつきがある場合、先行するパイプライン長の長い命令の実行が終了しないと、後続の命令の実行が終了していても処理を完了することができず、このため、リソースの解放が遅れ、それが原因でリソース溢れによるストールが起こる場合があるという問題点を有している。

【0006】本発明の目的は、前記従来技術の問題点を解決し、アウトオブオーダ実行を行うパイプライン計算機に対して命令列のディスパッチから命令の実行開始までの時間及び命令の実行終了からコンプリートまでの時間がなるべく短くなるように、すなわち、各命令がハードウエアリソースを占有する時間（ディスパッチから完了までの時間）がなるべく短くなるようにスケジュールされた命令列を得ることができる命令スケジューリング方法を提供することにある。

【0007】また、本発明の目的は、演算器数、命令毎の実行時間、ユースレイテンシ等のプロセッサに関する情報をテーブル形式（マシン記述テーブル）で入力するプロセッサに依存しないタイプのコンパイラにおいて、アウトオブオーダ実行の効果を考慮したスケジューリングを簡単に行うことができる命令スケジューリング方法を提供することにある。

【0008】

【課題を解決するための手段】本発明によれば前記目的は、命令列のアウトオブオーダ実行を行うパイプライン計算機の目的プログラムを生成するコンパイラにおける命令スケジューリング方法において、命令を実行する複数の演算器のパイプライン長がそれぞれ異なっている場合に、先行命令のターゲットデータをパイプライン長の長い後続命令がユースする場合のユースレイテンシを実際より大きく設定することにより、パイプライン長の長い命令のディスパッチを遅らせ、パイプライン長の長い命令をアウトオブオーダ実行させるようにスケジューリングすることにより達成される。

【0009】本発明は、前述の手段を備え、パイプライン長の長い命令のディスパッチのタイミングを遅らせてスケジュールすることにより、パイプライン長の長い命令のアウトオブオーダ実行を誘発し、後続命令のコンプリートのタイミングを早めることが可能になる。この結果、本発明によれば、ハードウエアリソースの使用効率を高めることができる。

【0010】

【発明の実施の形態】以下、本発明による命令スケジューリング方法の一実施形態を図面により詳細に説明する。

【0011】図1は本発明の一実施形態による命令スケ

ジューリング方法の処理手順を説明するフローチャート、図2はコンパイラの構成とカーネル生成処理の概略を説明する図、図3はアウトオブオーダ実行のプロセッサの構成の概略と命令処理の概略とを説明する図、図4はプロセッサ記述テーブルを説明する図、図5は具体例としての原始プログラムと変換された命令列とを説明する図、図6は図5に示す命令列に対するデータ依存性グラフを示す図、図7は図5に示す命令列に対するスケジュール後の命令の実行開始想定時間を説明する図、図8は図5に示す命令列に本発明によるスケジューリング処理を施した場合の命令列の実行の様子を説明する図、図9は図5に示す命令列に実施形態によるスケジューリング処理を施し場合の命令列の実行の様子を説明する図である。図2～図4において、7は原始プログラム、8はコンパイラ本体、9は構文解析部、10はコード生成部、11は最適化処理部、12は目的プログラム、13は命令フェッチ部、14は命令ディスパッチ制御部、15～18は演算器、19は命令コンプリート制御部、20、23はプロセッサ記述テーブルである。

【0012】まず、図2を参照してコンパイラの構成と処理の流れを説明する。コンパイラ8は、構文解析部9と、本発明によるスケジューリング処理を実行する最適化処理部11を備えるコード生成部10とにより構成されている。そして、コンパイラ8に入力される原始プログラム7は、構文解析部9で処理をされた後、コード生成部10に入力される。このコード生成部10内の最適化処理部11は、従来から行われている最適化処理の他に本発明によるスケジューリング処理を実行するものである。構文解析されたプログラムは、最適化処理部11内で本発明によるスケジューリング処理を受けることにより、演算の処理効率を向上させることができる配列を持った目的プログラム12に変換されて出力される。

【0013】スーパスカラ・アウトオブオーダ実行のプロセッサは、その概略を図3に示すように、コンパイラ8によりスケジューリング処理を受けた目的プログラム12を読み込む命令フェッチ部13と、演算部に命令を分配する命令ディスパッチ制御部14と、命令を並列に実行する複数の演算器15～18と、全命令の実行終了を確認して、リソースの開放を行う命令コンプリート制御部19とにより構成される。なお、演算器15～18としては、分岐演算器、浮動点演算器、固定点演算器、ロードストア演算器等の各種の演算器が用意されている。これらの演算器のそれぞれは、演算機能に対応して異なるステージ数の複数の演算ステージを有するパイプライン方式により構成されており、与えられた命令を並列に実行する。

【0014】前述のように構成されるプロセッサにおいて、コンパイラ8により生成された目的プログラム12は、命令フェッチ部13により順次読み込まれる。命令ディスパッチ制御部14は、命令列の順すなわちインオーダで、必要なハードウエアリソースを確保の上、命令に対応する演算器15～18に命令をディスパッチする。各演算器15～18は、ソースデータが得られ次第、元の命令列での順序とは無関係に、すなわちアウトオブオーダで与えられた命令を実行する。各演算器15～18での命令実行が終了すると、命令コンプリート制御部19は、命令列での順序上先行する全ての命令の実行が終了したことを確認し、インオーダでコンプリートしハードウエアリソースを解放する。

【0015】なお、図3には示していないが、命令フェッチ部13、命令ディスパッチ制御部14、複数の演算器15～18のそれぞれは、それらの入力側にキューバッファが設けられており、キューバッファ内に格納された命令を順番に取り出して処理を実行する。

【0016】本発明の一実施形態によるスケジューリング処理は、図2に示す最適化処理部11内で、図1に示すような処理手順により実行される。以下、この処理動作を具体的な例を上げて説明する。なお、以下に説明する本発明の一実施形態は、4命令同時ディスパッチ可能、ロード及びストア命令を2命令同時実行可能、かつ、浮動点演算命令を2命令同時実行可能なある種のスーパスカラプロセッサを想定する。

【0017】いま、例として、図5（a）に示す命令26～命令30からなるプログラムを考える。このプログラムは、コンパイラのコード生成部10において、図5（b）に示すような命令31～命令50による命令列に置き換えられる。命令26～命令30は、それぞれ加算命令であり、4命令の命令列に置き換えられるが、命令26が置き換えられた命令列30～34の意味を簡単に説明する。

【0018】命令26は、2つのデータY1、Z1の加算命令であり、この命令は、Y1の値をメモリから読み出してレジスタf1に格納するロード命令LD1（命令31）、Z1の値をメモリから読み出してレジスタf2に格納するロード命令LD2（命令32）、レジスタf1、f2の値を加算してレジスタf3に格納する加算命令ADD3（命令33）、レジスタf3の値をメモリのX1のアドレスに格納するストア命令ST3（命令34）に置き換えられる。命令27～30についても、それぞれ同様に置き換えられる。

【0019】図5（b）に示すように置き換えられた命令列は、図2に示す最適化処理部11に送られる。そして、最適化処理部11において、図1に示すスケジューリング処理が施される。

【0020】（1）まず、図4（a）に示すプロセッサ記述テーブル20を読み込む（ステップ1）。プロセッサ記述テーブル20は、演算器の種類・数等を記述するテーブル21と、命令毎の実行時間、ユースレイテンシを記述するテーブル22とにより構成される。そして、テーブル21には、図示例の場合、プロセッサが、ロー

ド命令ＬＤ、ストア命令ＳＴを実行する演算器ＡＵを２つ備え、２命令を同時に実行可能であり、また、加算命令ＦＡＤＤを実行する演算器ＦＵを２つ備え、２命令を同時に実行可能であることが示されている。また、テーブル２２には、図示例の場合、各命令の命令パイプラインの長さに対応する実行時間（マシンサイクル数）と、後続命令のユースレイテンシとが記述されている。

【００２１】そして、本発明の実施形態は、後続命令の実行時間が長い場合、その命令のユースレイテンシを、実際のプロセッサ動作を反映する従来技術の場合より増分させている。すなわち、後続する実行時間が長い命令のユースレイテンシ、例えば、ＬＤ命令に続いて実行されるＦＡＤＤ命令と、ＦＡＤＤ命令に続いて再度実行されるＦＡＤＤ命令とのユースレイテンシを従来技術による場合より１サイクルだけ増分している。

【００２２】これについて具体例で説明すると、図４（ｂ）に示す実際のプロセッサ動作を反映する従来技術の場合のプロセッサ記述テーブル２３は、演算器の種類・数等を記述するテーブル２４の内容が、図４（ａ）における本発明の実施形態のテーブル２１の内容と同一であり、命令毎の実行時間、ユースレイテンシを記述するテーブル２５の各命令の実行時間がテーブル２２と同一であっても、後続する実行時間が長い命令のユースレイテンシは、その前の命令の実行時間に重なるように設定されている。

【００２３】前述の図４（ｂ）において、ＦＡＤＤ命令に続いて再度ＦＡＤＤ命令が実行される場合、後続のＦＡＤＤ命令のユースレイテンシが、ＦＡＤＤ命令の実行時間が６サイクルであるのに対して４サイクルに設定されているのは、前の命令の実行時間の最後の２サイクルに、後続の命令の実行を重複して行うことができるからである。同様に、ＬＤ命令に続いてＦＡＤＤ命令が実行される場合、後続のＦＡＤＤ命令のユースレイテンシは、ＬＤ命令の実行時間の５サイクルより１つ短い４サイクルに設定されている。

【００２４】これに対して、本発明の実施形態は、ＬＤ命令に続いてＦＡＤＤ命令が実行される場合、ＬＤ命令の実行時間が５サイクルであるのに対して、その後に実行されるＦＡＤＤ命令に対するユースレイテンシを、ＬＤ命令の実行時間と同一の５サイクル（従来技術の場合より１サイクル多い）に設定し、ＦＡＤＤ命令に続いて再度ＦＡＤＤ命令が実行される場合、前の命令であるＦＡＤＤ命令の実行時間が６サイクルで、実行時間の最後のサイクル５で、後続の命令を開始させている。この場合にも、ユースレイテンシは、従来技術の場合より１サイクル多い値に設定されていることになる。

【００２５】なお、前述した図４（ａ）、図４（ｂ）において、ユースレイテンシに“０”が設定されているのは、命令相互間にデータの依存性がないことを示している。従って、この場合、データとリソースとが存在すれば、後続の命令を何時でも実行させることができる。

【００２６】（２）次に、図６に示すデータ依存性グラフを得る（ステップ２）。図６に示すデータ依存性グラフは、図５（ｂ）に示す命令列の４命令の組のそれぞれにおけるデータの依存性を示すもので、図６における有方向エッジに付随する数字は、命令間のユースレイテンシを示している。

【００２７】（３）次に、通常のリストスケジューリングにより命令のスケジューリングを行うスケジューリング本体処理を行う。この処理によりスケジューリングされた命令列は、命令の実行終了時間やコンプリートタイミングについての考慮はなされていない（ステップ３）。

【００２８】（４）ステップ３でスケジューリングされた命令列は、次回スケジュール候補命令選出処理に加えられ、命令列からソースデータが計算済である命令の集合が選出され、さらに、次回スケジュール命令決定処理において、実行スロット等の割り当てを行って、次サイクルに実行させる命令を決定する（ステップ４、５）。

【００２９】（５）全命令のスケジューリングが終了したか否かをチェックし、終了していれば処理を終了し、そうでない場合、ステップ４に戻って、ステップ４からの処理を繰り返す（ステップ６）。

【００３０】前述した処理により図５（ｂ）に示す命令列をスケジューリングした結果、各命令の実行開始想定時間は、図７（ａ）に示すようになる。本発明の一実施形態の場合、ロード命令に続いて実行される加算命令のユースレイテンシを図４（ａ）により説明したように、マシンサイクル数５に設定しているので、図５（ｂ）に示す加算命令３３は、第６サイクル（以降であれば何時でもよい）に実行が開始されることになる。

【００３１】そして、図７（ａ）に示したスケジュールの状況は、コンパイラによって作成されるもので、実際の処理装置での処理とは、ハードウエアリソースの不足、衝突とうの原因により一致するものではないが、最終的に処理装置で実行される命令列及び処理装置でのディスパッチ（Ｄ）・実行（Ｅ）・コンプリート（Ｃ）の処理は、図８に示すように実行される。

【００３２】図８から判るように、本発明の実施形態におけるパイプラインの長いＦＡＤＤ命令の実行終了待ちによるＳＴ命令のコンプリーションの遅れは、命令１０１について０サイクル、命令１０３について１サイクル、命令１０５について１サイクル、命令１０７について２サイクル、命令１０９について２サイクルの合計６サイクルに抑えることができる。

【００３３】比較のため、図５（ｂ）に示す命令列を、従来技術の場合と同様にパイプラインの長短を考慮することなくスケジューリングした場合のの各命令の実行開始想定時間は、図７（ｂ）に示すようになる。図７（ｂ）から判るように、ユースレイテンシの差を反映し

て、ＡＤＤ命令の実行開始時間が、本発明の実施形態の場合よりも１サイクル早くなっている。そして、最終的に実際の処理装置で実行される命令列及びそのディスパッチ・実行・コンプリートの処理は、図９に示すように実行される。

【００３４】図９から判るように、従来技術の場合、ＡＤＤ命令の実行終了が遅いために、後続のＬＤ命令、ＳＴ命令のコンプリートの遅れは合計９サイクルとなり、本発明と比べ３サイクル分無駄にハードウエアリソースを消費していることになる。

【００３５】前述した本発明の実施形態によれば、各命令ディスパッチから実行開始までの時間及び実行終了からコンプリートまでの時間を短くおさえ、各命令がハードウエアリソースを占有する時間が短くなるように命令列をスケジュールすることができ、これにより、ハードウエアリソースの効率的な利用を可能にして、プログラム全体の処理効率を向上させることができる。

【００３６】本発明の実施形態を説明するために例として用いたプログラムは、小さなものであるため、図８、図９から本発明の実施形態による効果が見にくいが、前述した本発明の実施形態によれば、プログラムが大きくなった場合に、ハードウエアリソースの効率的な利用による効果が顕著なものとなり、プログラム全体の処理効率を向上させることができる。

【００３７】

【発明の効果】以上説明したように本発明によれば、従来のスケジューリング法におけるプロセッサ記述テーブルのユースレイテンシの値を操作してアウトオブオーダ実行を意図的に引き起こすことにより、命令パイプライン長が一定でない場合に予想されるコンプリートの遅延によるハードウエアリソースの浪費を防止することを可能にして、プログラム全体の処理効率を向上させることができる。

【図面の簡単な説明】

【図１】本発明の一実施形態による命令スケジューリング方法の処理手順を説明するフローチャートである。

【図２】コンパイラの構成とカーネル生成処理の概略を説明する図である。

【図３】アウトオブオーダ実行のプロセッサの構成の概略と命令処理の概略とを説明する図である。

【図４】プロセッサ記述テーブルを説明する図である。

【図５】具体例としての原始プログラムと変換された命令列とを説明する図である。

【図６】図５に示す命令列に対するデータ依存性グラフを示す図である。

【図７】図５に示す命令列に対するスケジュール後の命令の実行開始想定時間を説明する図である。

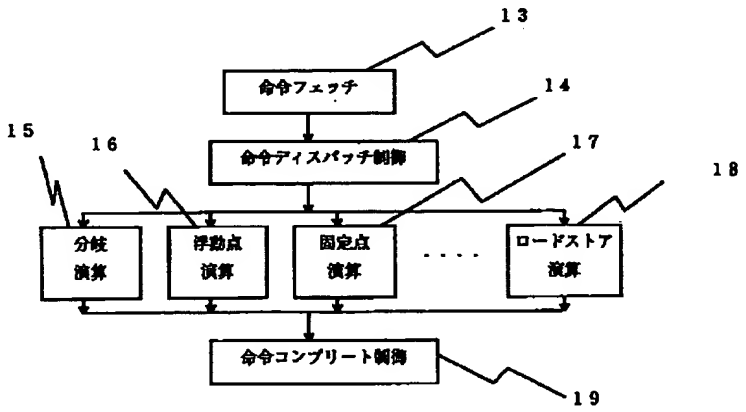
【図８】図５に示す命令列に本発明によるスケジューリング処理を施した場合の命令列の実行の様子を説明する図である。

【図９】図５に示す命令列に実施形態によるスケジューリング処理を施し場合の命令列の実行の様子を説明する図である。

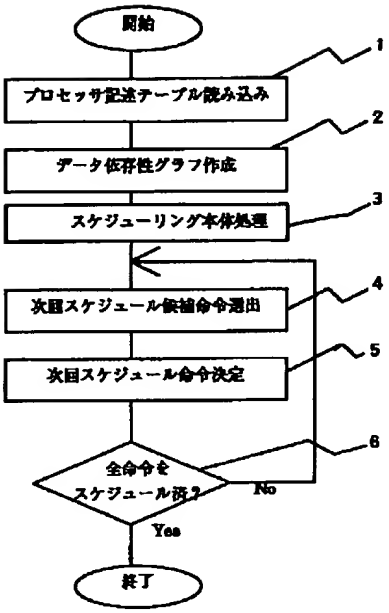
【符号の説明】

７　原始プログラム
８　コンパイラ本体
９　構文解析部
１０　コード生成部
１１　最適化処理部
１２　目的プログラム
１３　命令フェッチ部
１４　命令ディスパッチ制御部
１５～１８　演算器
１９　命令コンプリート制御部
２０、２３　プロセッサ記述テーブル

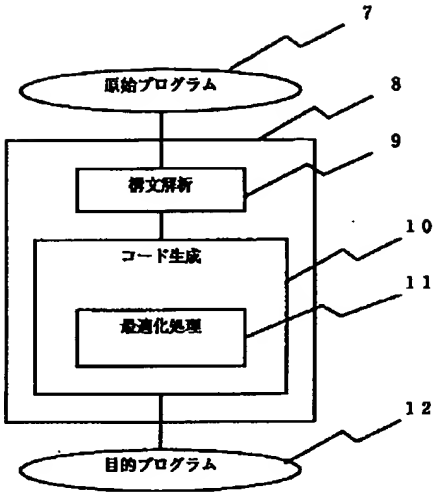
【図３】



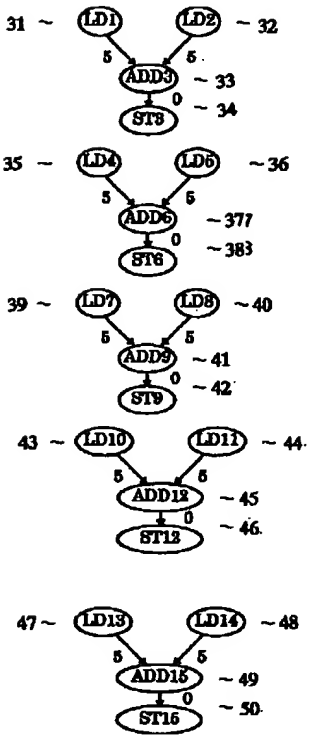
【図1】


【図1】
開始
プロセッサ記述テーブル読み込み
データ依存性グラフ作成
スケジューリング本体処理
次回スケジュール候補命令選出
次回スケジュール命令決定
全命令を
スケジュール済？
No
Yes
終了
1
2
3
4
5
6

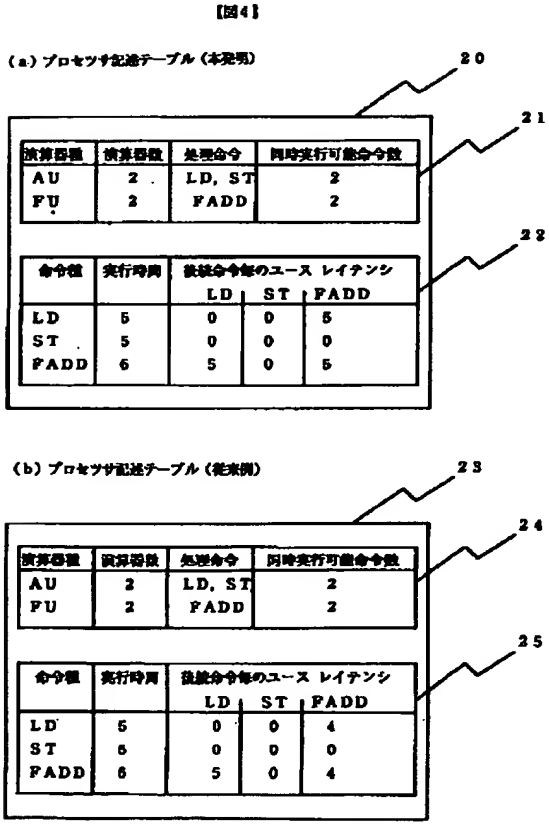
【図2】


【図2】
原始プログラム
構文解析
コード生成
最適化処理
目的プログラム
7
8
9
10
11
12

【図6】


【図6】
LD1
LD2
ADD3
ST3
LD4
LD5
ADD6
ST6
LD7
LD8
ADD9
ST9
LD10
LD11
ADD12
ST12
LD13
LD14
ADD15
ST15
31
32
33
34
35
36
377
383
39
40
41
42
43
44
45
46
47
48
49
50
5
0

【図4】


【図4】
（a）プロセッサ記述テーブル（本発明）
演算器種
演算器数
処理命令
同時実行可能命令数
AU
2
LD, ST
2
FU
2
FADD
2
命令種
実行時間
後続命令毎のユース レイテンシ
LD
ST
FADD
LD
5
0
0
5
ST
5
0
0
0
FADD
6
5
0
5
20
21
22
（b）プロセッサ記述テーブル（従来例）
演算器種
演算器数
処理命令
同時実行可能命令数
AU
2
LD, ST
2
FU
2
FADD
2
命令種
実行時間
後続命令毎のユース レイテンシ
LD
ST
FADD
LD
5
0
0
4
ST
5
0
0
0
FADD
6
5
0
4
23
24
25

【図5】

【図5】

(a)原始プログラム

26～	X1=Y1+Z1
27～	X2=Y2+Z2
28～	X3=Y3+Z3
29～	X4=Y4+Z4
30～	X5=Y5+Z5

(b)最適化前の命令列

31～	LD1:LD	f1, +0x10(r10)
32～	LD2:LD	f2, +0x10(r12)
33～	ADD3:FADD	f3, f1, f2
34～	ST3:ST	f3, +0x10(r11)
35～	LD4:LD	f4, +0x10(r10)
36～	LD5:LD	f5, +0x10(r12)
37～	ADD6 FADD	f6, f4, f5
38～	ST6:ST	f6, +0x10(r11)
39～	LD7:LD	f7, +0x10(r10)
40～	LD8:LD	f8, +0x10(r12)
41～	ADD9:FADD	f9, f7, f8
42～	ST9:ST	f9, +0x10(r11)
43～	LD10:LD	f10, +0x10(r10)
44～	LD11:LD	f11, +0x10(r12)
45～	ADD12:FADD	f12, f10, f11
46～	ST12:ST	f12, +0x10(r11)
47～	LD13:LD	f13, +0x10(r10)
48～	LD14:LD	f14, +0x10(r12)
49～	ADD15:FADD	f15, f14, f13
50～	ST15:ST	f15, +0x10(r11)

【図7】

【図7】

（a）本発明によるスケジューリング

実行開始時間	実行命令
1	LD1、LD2
2	LD4、LD5
3	LD7、LD8
4	LD10、LD11
5	LD13、LD14
6	ADD3、ST3
7	ADD6、ST6
8	ADD9、ST9
9	ADD12、ST2
10	ADD15、ST15

（b）従来技術によるスケジューリング

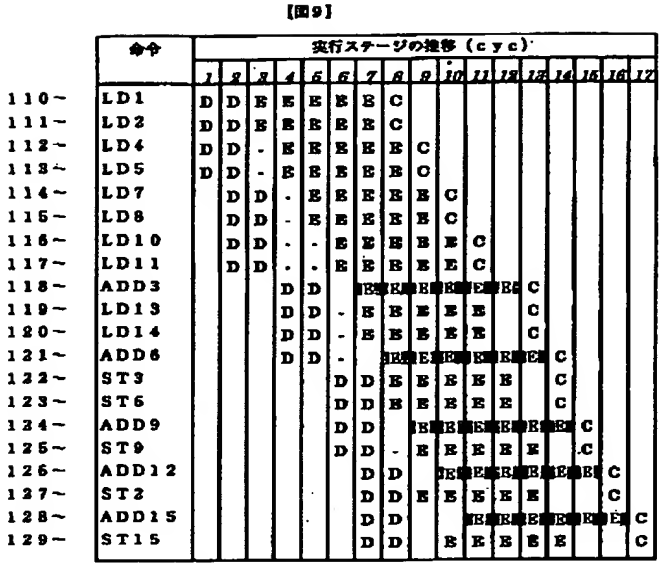
実行開始時間	実行命令
1	LD1、LD2
2	LD4、LD5
3	LD7、LD8
4	LD10、LD11
5	ADD3、LD13、LD14
6	ADD6、ST3、ST6
7	ADD9、ST9
8	ADD12、ST2
9	ADD15、ST15

【図8】

【図8】

	命令	実行ステージの遷移（cyc）															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
90～	LD1	D	D	E	E	E	E	E	C									
91～	LD2	D	D	E	E	E	E	E	C									
92～	LD4	D	D	-	E	E	E	E	E	C								
93～	LD5	D	D	-	E	E	E	E	E	C								
94～	LD7		D	D	-	E	E	E	E	E	C							
95～	LD8		D	D	-	E	E	E	E	E	C							
96～	LD10		D	D	-	-	E	E	E	E	E	C						
97～	LD11		D	D	-	-	E	E	E	E	E	C						
98～	LD13				D	D	-	E	E	E	E	E	C					
99～	LD14				D	D	-	E	E	E	E	E	C					
100～	ADD3				D	D		E	E	E	E	E	E	C				
101～	ST3				D	D	-	-	E	E	E	E	E	C				
102～	ADD6						D	D	E	E	E	E	E	E	C			
103～	ST6						D	D	E	E	E	E	E		C			
104～	ADD9						D	D		E	E	E	E	E	E	C		
105～	ST9						D	D	-	E	E	E	E	E		C		
106～	ADD12							D	D		E	E	E	E	E	E	C	
107～	ST2							D	D	E	E	E	E	E			C	
108～	ADD15							D	D			E	E	E	E	E	E	C
109～	ST15							D	D	-	E	E	E	E	E			C

【図9】



【図9】

	命令	実行ステージの推移（cyc）															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
110～	LD1	D	D	E	E	E	E	E	C									
111～	LD2	D	D	E	E	E	E	E	C									
112～	LD4	D	D	-	E	E	E	E	E	C								
113～	LD5	D	D	-	E	E	E	E	E	C								
114～	LD7		D	D	-	E	E	E	E	E	C							
115～	LD8		D	D	-	E	E	E	E	E	C							
116～	LD10		D	D	-	-	E	E	E	E	E	C						
117～	LD11		D	D	-	-	E	E	E	E	E	C						
118～	ADD3				D	D		E	E	E	E	E	E	E	C			
119～	LD13				D	D	-	E	E	E	E	E			C			
120～	LD14				D	D	-	E	E	E	E	E			C			
121～	ADD6				D	D	-		E	E	E	E	E	E	E	C		
122～	ST3						D	D	E	E	E	E	E		C			
123～	ST6						D	D	E	E	E	E	E		C			
124～	ADD9						D	D		E	E	E	E	E	E	C		
125～	ST9						D	D	-	E	E	E	E	E		C		
126～	ADD12							D	D		E	E	E	E	E	E	C	
127～	ST2							D	D	E	E	E	E	E			C	
128～	ADD15							D	D			E	E	E	E	E	E	C
129～	ST15							D	D		E	E	E	E	E			C